DESIGN AND IMPLEMENTATION OF BRPY: A BIORECOGNITION SYSTEM (FACE DETECTION AND RECOGNITION, AGE ESTIMATION AND GENDER ESTIMATION) BASED ON THE FACE.

Iván de Paz Centeno, Eduardo Fidalgo, Oscar García-Olalla, Enrique Alegre
Dpto. Ingeniería Eléctrica y de Sistemas y Automática, Universidad de León, Campus de Vegazana s/n,
24071 León, Spain,
ipazce00@estudiantes.unileon.es

ABSTRACT

This document exposes the design and implementation of BRPY, an open source multi platform program oriented to the face detection and recognition, age and gender estimation based on the face, in real time, programmed in python and C++ over OpenBR and OpenCV. This document also focuses the hardware appliance involved in the execution of the core, being the ODroid U3 the targeted platform; and the design and implementation of the program interface.

Keywords: openbr, age estimation, gender estimation, detection, recognition, face, algorithm, python, BRPY.

1 INTRODUCTION

One of the most common tasks executed nearly 6000 billion times per second in the world is the face recognition process. Also, the age estimation and gender estimation may fall in the same group. These tasks are usually underestimated by the humans, which are, until the recent times, the only *appliance* capable of executing them.

The exponential evolution of the hardware, the techniques and the algorithms developed to accomplish these tasks has led us the needed platform to perform them in a real-time context, even on an embedded system of the size of ODroid U3, which may fit in a box whose base has the dimensions of a credit card.

Furthermore, the recent arise of new vision and biorecognition frameworks like OpenCV and OpenBR, added layers of abstraction enough to develop applications that use these tasks as a feasible approach to solve complex vision problems. This is the case for BRPY, the acronym for ***B**io**R**ecognition on **PY**thon* project, which enhances the most common and complex tasks in the world in an easy implementation, using a camera in a streaming recording state to process and apply the algorithms, showing the results in nearly real-time.

In the last decades, several research groups have been working in face detection and recognition algorithms. Texture descriptors have demonstrated their high capabilities to extract relevant features [2,4]. One of the most used algorithms was proposed by Ojala et al: Local Binary Pattern (LBP) [7]. Due to its simplicity and well description discrimination, lot of methods based on it have been published in the last years [1,3].

The utilized materials to accomplish this project are shown in section 2. The software and hardware layers are explained in the section 3. Experiments are briefly described in section 4 and final conclusions are summarized in section 5.

2 MATERIALS

2.1 APPLIANCE

Because of the versatility of the program, the appliance involved in its execution may depend on the configuration taken for the project. For the purpose of this document, the core and the interface will be executed by the same device, but it's not limited to that configuration. This is extensively explained in the section 3.

2.1.1 ODROID U3

Figure 1: ODroid U3.

The ODroid U3 is a low-priced high-featured device from Hardkernel, faster by far than a Raspberry Pi2 to perform the algorithm computations needed by this project.
Although it has the same number of cores than a Raspberry Pi2, its clock frequency is approximately two times higher than the Raspberry Pi2 ones, as shown in Table 1:

Table 1: ODroid U3 specifications. [5]

ODroid U3 specifications.	
CPU	Quad core ARM Cortex-A9 @ 1.7 GHz
GPU	Mali-400 QuadCore at 440 MHz
RAM	2 Gbyte LP-DDR2 800 MHz
Network	10/100 Mbps Ethernet
Storage	EMMC4.5 HS200 Flash Storage and MicroSD Card slot.
USB	USB2.0 host x 3 and MicroUSB x 1
Other	40 PIN GPIOs. SPI. I2C. MicroHDMI.
Supported OS	Android 4.x+, Ubuntu 13.x+
Power	5V 2A

The BRPY core runs smooth under this appliance, processing nearly 5 frames per second at a 640x480 resolution of a USB camera without optimizations, using only one of the four available cores. This means that this device executes the face detection, face recognition, age estimation and gender estimation 5 times per second, and is capable of multiply that speed by 4 on a full CPUs dedication.

2.1.2 CLIENT DEVICE

Since the displayed interface of the application is generated dynamically by a web application, it might run on any device that is capable of process and interpret HTML5, Javascript and also has a connectivity (and rights) to access the host (usually, the one that runs the core). Any modern web browser may be able to display the interface and allow the user to monitor and visualize the program. This means that the interface can be shown in any kind of computer, even in a mobile phone.

3 SOFTWARE AND HARDWARE LAYERS

The nature of the BRPY is to distribute computation and roles of processing amongst different devices, leading to a distributed system schema. It is divided into three roles or layers which can work isolated or grouped into different devices or same device, respectively; however they must keep a connectivity with each other.

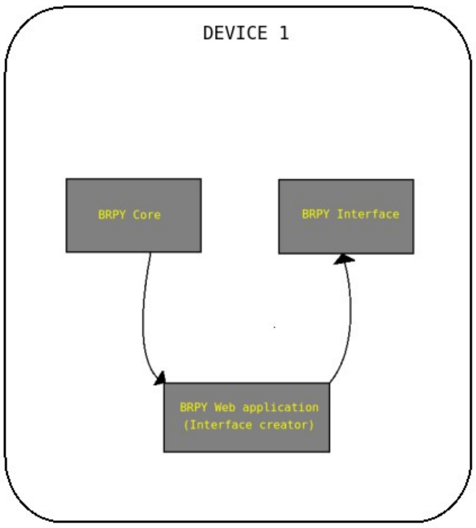

Figure 2: Basic BRPY topology: one device to process the three layers.

The layers are software components but they can be isolated into hardware components. This versatility allows us to use a range of devices that can go from a single device to do the whole work (figure 2), to three (or more) devices to distribute the work (figure 3). This document covers the first case, in which only one device is needed to process the camera and display the interface.

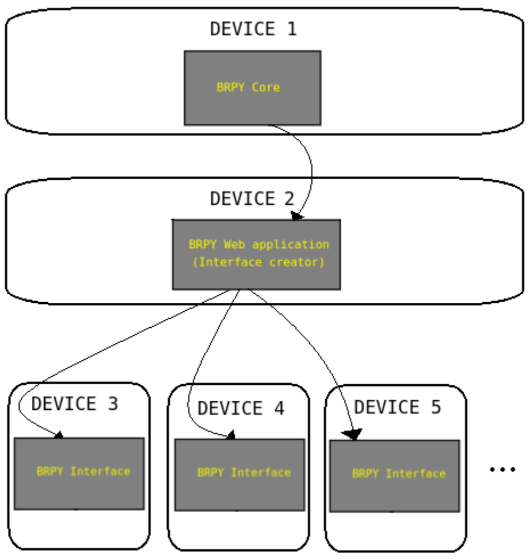

Figure 3: Most complex BRPY topology: one device to capture and process the frames, other to generate and serve interfaces and other(s) to display the

3.1 THE CORE LAYER

The core is the layer which performs the capture from a given video device, the processing of each of its frames to detect faces, recognize them, estimate its age and its gender, and store the results in the BRPY Web Application layer. This is accomplished thanks to the OpenBR framework, which provides efficient algorithms to perform biorecognition on pictures.

OpenBR [8] is a framework developed to extend functionalities of OpenCV [9]. This means that its underlying core is OpenCV. It also simplifies the access and processing of pictures using a new interpreted language in order to apply algorithms to templates. The templates are the basic containers of the algorithms results. To know more about OpenBR, it's highly encouraged to take a look at its paper [6].

BRPY has a built-in wrapper for OpenBR called PYOBR (**PY**thon **O**pen**BR**) which wraps its basic functionality to be utilized from a Python environment. The implementation of this wrapper is *object-oriented*, making it easy to apply basic biorecognition algorithms available at OpenBR 0.50 to pictures and to retrieve the results (equally to OpenBR, as templates).

For the detection of faces, its underlying algorithm is the Viola-Jones[10], provided by the OpenCV framework. This algorithm restricts BRPY to detect only frontal faces, which will be the base to apply age and gender estimation, and face recognition algorithms.

For age and gender estimation, its underlying algorithms are based in SVM, trained by the features of the 4SF[6].

For recognition purposes, BRPY is able to learn new faces at the time they are being detected if they are not actually recognized. This behaviour allows us to generate a dataset of the people using the program, granting the possibility to recognize that person the next time it's detected on the camera. For the recognition process, its underlying algorithm is a 4SF (Spectrally Sampled Structural Subspaces Features), based on LBP and dense SIFT[6].

The results of the application are stored in a database which serves as a link between the core and the web application layer.

3.2 THE WEB APPLICATION LAYER

This layer generates the application interface on demand, according to the HTTP requests being done by the user/s. It exposes an API written in PHP and served by a Web Server (like Apache2 or Nginx). The interface layer communicates with this layer by a polling method, requesting data at a fixed rate (nearly 5 FPS).

The API of this layer processes the database in order to retrieve the data and convert it to an XML format, understandable by the interface.

For the purposes of this document and for security reasons, the web application is limited to serve only in an operating system scope, not attending to external or local connections since all the layers are present in the same device.

3.3 THE INTERFACE LAYER

The interface layer is developed to continuously poll the web application layer and request information of the current detections. This information is retrieved in an XML format, which needs to be processed by the underlying JavaScript interface handler and represented accordingly into the view area of the browser. The result can be seen in the figure 4.

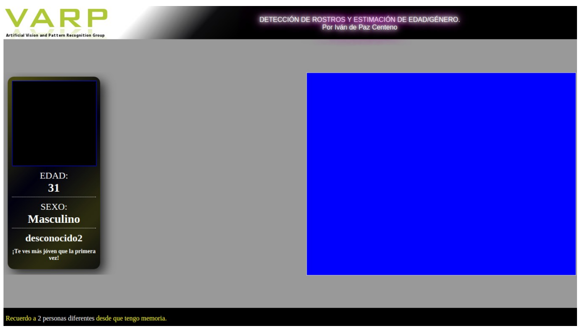
VARP
Artificial Vision and Pattern Recognition Group
DETECCIÓN DE ROSTROS Y ESTIMACIÓN DE EDAD/GÉNERO.
Por Iván de Paz Centeno
EDAD:
31
SEXO:
Masculino
desconocido2
¡Te ves más jóven que la primera vez!
Recuerdo a 2 personas diferentes desde que tengo memoria.

Figure 4: Example of the view from the interface.

The right blue box represents the current frame being processed and the left black box represents a face detected in the frame. Since the core is capable of detect multiple faces in the same frame, the result of the detections may increase or decrease the amount of detection boxes from the interface.

The rendering of the interface is done by the user's device. As explained in the section 3, for the purpose of this document, the processing is completely done by the ODroid U3, serving as a core, web application and interface renderer, all at the same time.

3.3 THE PYOBR WRAPPER

PyOBR wrapper is written to facilitate the communication between the python core program and the OpenBR framework. It's designed in an Object Oriented Schema, in python; and like OpenBR, it's also template-oriented.

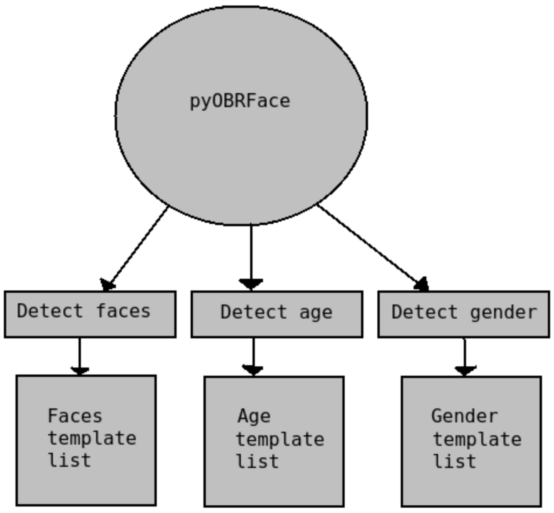

Figure 5: Example of basic algorithms accessible by the PyOBRFace.

Its main class is the *pyOBRFace*, a class that exposes the algorithms allowed by faces pictures to be executed once loaded a picture into the class, as shown in figure 5.

The call to the methods are parameter-less and returns a list of templates as a result: if the processed picture holds more than one face, each of the processed faces will generate a unique template and will be stored in the list.

The template in OpenBR has a correspondence in the PyOBR: the *PyOBRTemplate*. This class holds all the metadata result of an applied algorithm. The access to this metadata must be demuxed by using the corresponding PyOBR template wrapper, as shown in figure 6.

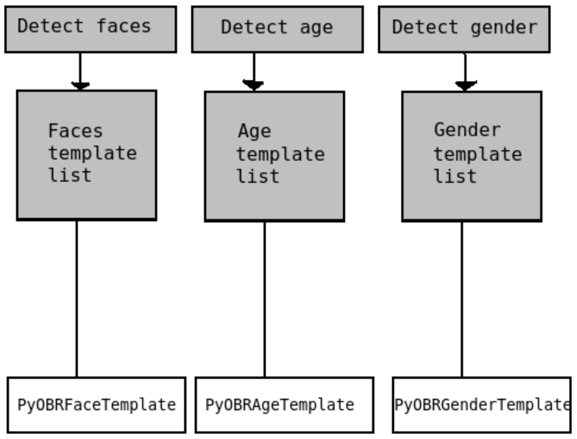

Figure 6: Different templates available to wrap the PyOBRTemplate.

These template wrappers access to the metadata segment according to the kind of template that is supposed to be:

- The *PyOBRFaceTemplate* has methods to retrieve characteristics of the face detection, like the face position or the number of confident faces.
- The *PyOBRAgeTemplate* has methods to retrieve characteristics from an age estimation, that is, the age as a float value.
- The *PyOBRGenderTemplate* has methods to retrieve characteristics from a gender estimation, that is, the gender value as an integer (0 = male, 1 = female).

Each of the algorithms exposed by the OpenBR for faces will have a template wrapper available to parse its result.

4 EXPERIMENTS AND RESULTS

While the application was being developed, for demonstration purposes, was tested with a few faces captured by a USB camera in a real time context. Though the face detection algorithm worked quite good in almost every environment, the experienced behaviour of the estimation algorithms showed the drawbacks of having a badly illuminated room, which resulted in an incorrect age estimation (the most accurately estimation was the gender).

BRPY hasn't been tested against datasets yet, however the accuracy of the application is subject to the accuracy of the underlying algorithms, mentioned in the OpenBR paper[6].

5 CONCLUSIONS

The design of BRPY implied the design of PyOBR, which implied the study of the OpenBR behaviour. Although the OpenBR is a layer of abstraction to complex algorithms and OpenCV, the PyOBR is another layer to abstract the access to these algorithms through OpenBR. Each of the abstraction layers leads the developer to an easier interface (which is the final objective of the abstraction), however, the number of applied layers may decrease the speed of the application, which might not be the best approach for an embedded system like the used for the coverage of this document. Fortunately, the exponential increase in the computers capacity sends this problem to a background layer, bringing to front the possibility to develop applications fast and easy using scripting languages (like python is) and avoiding the need to perform heavy optimizations to achieve the desired results.

Although these abstractions allow the development of simple applications, BRPY is not that simple, as stated in this document: it's utilizing multiple technologies and platforms to be executed, not limited to a single programming language. However, thanks to this approach, it is capable of running and being shown on any device and on any operating system, making it a really versatile program.

ACKNOWLEDGEMENTS

The development of BRPY has been partially financed by Cátedra de Telefónica de la Universidad de León, under the project “Sistemas portables para detección de rostros y estimación de edad y género” and via the predoctoral FPU fellowship program from the Spanish Government.

References

[1] García-Olalla, O.; Alegre, E.; Fernández-Robles, L.; Malm, P.; Bengtsson E. (2015) Acrosome integrity assessment of boar spermatozoa images using an early fusion of texture and contour description.

[2] García-Olalla, O.; Alegre, E.; Fernández-Robles, L.; Gónzalez-Castro, V. (2014) Local Oriented Statistical Information Booster (LOSIB) for texture classification. International Conference on Pattern Recognition.

[3] García-Olalla, O.; Alegre, E.; Fernández-Robles, L.; García-Ordás, M.T.; García-Ordás, D. (2013). Adaptive Local binary pattern with oriented standard deviation (ALBPS) for texture classification. EURASIP Journal on Image and Video Processing.

[4] González-Castro, V.; Alegre, E.; García-Olalla, O.; Fernández-Robles, L.; García-Ordás, M.T. (2012) Adaptive pattern spectrum image description using Euclidean and geodesic distance without training for texture classification. IET Computer Vision.

[5] Hardkernel, specifications of the ODroid U3, http://www.hardkernel.com/main/products/prdt_info.php?g_code=G138745696275
Last accessed: 23/06/2015

[6] Klontz, J.C.; Klare, B.F.; Klum, S.; Jain, A.K.; Burge, M.J., (2013) Open source biometric recognition, Biometrics: Theory, Applications and Systems (BTAS), IEEE Sixth International Conference on.

[7] Ojala, T.; Pietikainen, M.; Harwood, D. (1994). Performance evaluation of texture measures with classification based on kullback discrimination of distributions. Proceedings of the 12th IAPR International Conference on Pattern Recognition (ICPR).

[8] Openbiometrics.org , OpenBR webpage, http://www.openbiometrics.org
Last accessed: 24/06/2015

[9] Opencv.org, OpenCV webpage,
www.opencv.org
Last accessed: 24/06/2015

[10] Viola, P.; Jones, M. (2001). Rapid Object Detection using a Boosted Cascade of Simple Features, Conference on Computer Vision and Pattern Recognition.